On the merits of SVC-based HTTP Adaptive Streaming

Jeroen Famaey*, Steven Latré*, Niels Bouten*, Wim Van de Meerssche*,
Bart De Vleeschauwer†, Werner Van Leekwijck†, and Filip De Turck*

* Ghent University – IBBT
Department of Information Technology
Email: jeroen.famaey@intec.ugent.be
†Alcatel-Lucent Bell Labs, Copernicuslaan 50, B-2018 Antwerpen, Belgium

***Abstract*—HTTP Adaptive Streaming (HAS) is quickly becoming the dominant type of video streaming in Over-The-Top multimedia services. HAS content is temporally segmented and each segment is offered in different video qualities to the client. It enables a video client to dynamically adapt the consumed video quality to match with the capabilities of the network and/or the client's device. As such, the use of HAS allows a service provider to offer video streaming over heterogeneous networks and to heterogeneous devices. Traditionally, the H.264/AVC video codec is used for encoding the HAS content: for each offered video quality, a separate AVC video file is encoded. Obviously, this leads to a considerable storage redundancy at the video server as each video is available in a multitude of qualities. The recent Scalable Video Codec (SVC) extension of H.264/AVC allows encoding a video into different quality layers: by dowloading one or more additional layers, the video quality can be improved. While this leads to an immediate reduction of required storage at the video server, the impact of using SVC-based HAS on the network and perceived quality by the user are less obvious. In this article, we characterize the performance of AVC- and SVC-based HAS in terms of perceived video quality, network load and client characteristics, with the goal of identifying advantages and disadvantages of both options.**

I. Introduction

The popularity of multimedia services has grown phenomenally over the last decade. It has recently become the most prevalent type of traffic on the Internet and its share is expected to increase even further in the near future [1]. Recently, HTTP-based Adaptive Streaming (HAS) protocols (e.g., Microsoft Smooth Streaming, Apple Live Streaming, Adobe HTTP Dynamic Streaming and Dynamic Adaptive Streaming over HTTP – DASH) have started substituting traditional multimedia streaming technologies (e.g., Real-Time Transport Protocol). HAS splits video streams into temporal segments, which are offered at multiple qualities. The client application can independently decide which quality to request of each segment, allowing it to adapt to dynamic network conditions and device capabilities, in order to ensure a continuous viewing experience for the end-user. The use of HTTP as a transport protocol additionally allows HAS streams to easily traverse NATs and firewalls. Moreover, the existing HTTP delivery infrastructure can be seamlessly adopted.

The content encoding process of existing HAS solutions is based on Advanced Video Coding (AVC), which introduces significant amounts of content redundancy across quality levels [2]. This leads to increased storage and bandwidth requirements, as well as reduced caching efficiency. Recently, the combination of HAS with Scalable Video Coding (SVC) was proposed as a means to circumvent these disadvantages [2], [3]. SVC reduces the amount of content redundancy by letting each quality level depend on the previous one. The lowest quality level, called the base layer, can be decoded independently, while all higher levels can only be decoded in combination with the previous ones. Although the use of SVC reduces content redundancy across quality levels, it does introduce extra encoding overhead [4], which increases the total bit-rate of the content stream.

The aim of this paper is to investigate the merits of an SVC-based HAS solution, using detailed simulation results. Concretely, a state-of-the-art AVC-based HAS solution is compared to a novel SVC-based HAS implementation. The advantages and disadvantages of both approaches are studied as a function of the network conditions as well as the video characteristics. Additionally, the trade-off between AVC content redundancy and SVC encoding overhead is evaluated in detail.

The remainder of this paper is structured as follows. Section II gives an in-depth overview of research on state-of-the-art AVC-based HAS protocols, as well as novel research efforts on SVC-based HAS solutions. A formal overview of the client rate adaptation algorithms, implemented in the simulated prototype, is given in Section III. Subsequently, Section IV lists the simulation results and compares the AVC and SVC-based solutions. Finally, Section V concludes the paper.

II. Related work

Several hard- and software companies offer their own implementation of HAS (e.g., Microsoft's Smooth Streaming [1], Apple's HTTP Live Streaming [5], Adobe's HTTP Dynamic Streaming [2]). More recently, a standardized solution has been proposed by the Moving Pictures Experts Group (MPEG), called Dynamic Adaptive Streaming over HTTP [6]. While

[1] Smooth Streaming - The Official Microsoft IIS Site - http://www.iis.net/download/SmoothStreaming

[2] HTTP Dynamic Streaming - http://www.adobe.com/products/hds-dynamic-streaming.html

these different implementations each have their own syntactic differences in the protocol, they all adopt the same architectural design: a standard web server, offering the segmented HAS content, the transmission of the segments over standard HTTP connections and an intelligent video client who uses a quality selection heuristic to determine at which quality to download future segments. While HAS is fairly recent, it is increasingly being used for video streaming. De Cicco *et al* discuss how Akamai's Content Delivery Network (CDNs) uses HAS to stream video over HSDPA links [7]. For more information about HAS, we refer to [8].

To further improve the delivery of HAS-based streaming solutions, optimizations can be performed both in the network, at the server or at the client. Liu *et al* present an in-network optimization of HAS for 3GPP networks [9]. By parallelizing the download and request of HAS segments, a better resource utilization can be achieved. Similarly, Pu *et al* present a scheduling algorithm for HAS to improve the network utilization in CDNs [10]. A proxy on the last-hop wireless mile that alters the TCP protocol for optimized HAS-based delivery is proposed by the same authors in [11]. A new TCP variant is introduced for the last mile that can increase the average throughput in wireless environments. In this paper, we focus on modifications to traditional HAS at the server and client side.

Most server-side optimizations focus on a differentiated encoding scheme for HAS. Traditionally, the H.264/AVC codec is used for streaming video. The deployment of HAS requires to encode the source video into multiple independent AVC videos, each encoded in a different quality. Obviously, this leads to a large penalty in server-side storage as each video is now encoded and stored in multiple versions. The recent introduction of the Scalable Video Codec [12] extension of AVC allows alleviating this. The combination of SVC and HAS is therefore increasingly being investigated. The theoretic advantages and disadvantages of using SVC instead of AVC are discussed in Huysegems *et al* [2]. Advantages that are mentioned are a theoretically better play-out during fluctuations (as SVC always downloads the lowest quality first) and a reduction in storage and bandwidth requirements. However, they also identify important challenges for SVC such as the penalty in bitrate for encoding SVC and an increasing vulnerability to high round trip times. Sanchez *et al* discuss the benefits of using SVC for HAS delivery in terms of web caching and saved uplink bandwidth and propose a scheduling algorithm for live HAS delivery [13], [3]. Additionally, an initial comparison of SVC and AVC is carried out but focused around the observed live latency. Both previous studies use a rather straightforward SVC-based quality selection heuristic that does not fully exploit the advantages of SVC-based HAS. SVC-based client heuristics can decide to either download the next segment or increase the quality of previously downloaded segments: this is not taken into account in [2], [13], [3]. Furthermore, we target a broader and detailed comparison of SVC- and AVC-based HAS, taking into account different configurations, with the goal of deriving configuration guidelines.

While the traditional AVC-based HAS implementations typically already come with existing video client heuristics, a plethora of new video client heuristics have also been proposed in literature. These new heuristics either modify the heuristic to improve its applicability to a particular domain or exploit the advantages of SVC-based HAS. For example, Liu *et al* discuss a specific video client heuristic for CDNs [14], while Adzic *et al* present a specific client heuristic for mobile environments [15]. Schierl *et al* propose a generic algorithm for selecting the next video quality to download [16]. The selection is done using a priority-based scheme and, amongst others, applied to SVC. In this case, a higher priority is given to the base layer compared to the enhancement layers. Although the approach shows to increase the robustness against playback interruptions, the proposed algorithm is video codec agnostic. Therefore, more specific algorithmic decisions are possible for SVC specifically in the design of video client heuristics. Such specific decisions are taken into account in the SVC-based heuristic presented by Andelin *et al* [17]. This algorithm uses a slope to define the trade-off an algorithm has between downloading the next segment and upgrading a previously downloaded segment. As this algorithm is the state of the art in SVC-based heuristics, it is used in this paper to compare AVC- vs SVC-based. We discuss this algorithm in more detail in Section III.

III. Client rate adaptation algorithms

In this section, we discuss the algorithmic details of three different video client heuristics that are used for comparing the benefits between AVC- and SVC-based HAS. For AVC, we use the heuristic used by the Microsoft Smooth Streaming v1 implementation. For SVC, Microsoft's Smooth Streaming algorithm is adapted for SVC-based selection and a second heuristic is taken into account, originally proposed by Andelin *et al* [17], that allows exploiting the specifics of SVC-based HAS.

A. AVC: Microsoft's Smooth Streaming heuristic

The algorithm presented in this section is based on an open source version of the algorithm included in the Microsoft Smooth Streaming video player[3]. An overview of the Microsoft's Smooth Streaming heuristic is shown in Algorithm III-A. The heuristic continuously evaluates the status of the play-out buffer and makes its decision based on the comparison of the buffer's state with 3 thresholds: a lower threshold L, an upper threshold U and a panic threshold P. The goal of the heuristic is to maintain a steady state of the play-out buffer and download the highest quality possible. Every time a new segment is downloaded, the buffer's state is evaluated to decide on the next segment. The heuristics starts off in buffering mode. This means that it follows a more aggressive way of increasing the quality. It downloads the highest possible quality according to the performed throughput measurements (line 6). This behavior is overridden if the buffer is showing signs of decrease or slow changes. In this case, the quality can only be increased or decreased with one level (lines 4-5). This is to avoid oscillations in the decision of the heuristic: continuous fluctuations in the experienced quality of a video is known to be annoying to the user. The buffering state continues until the buffer is almost completely full (lines 7-8). In this case, the heuristic goes into a steady state.

[3]Original source code available from https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming/

In steady state, the change in quality is more conservative: the quality is only increased or decreased with one level at a time. To determine whether to change the quality, the heuristic analyses the buffer state, the download time of the previous segment and the potential occurrence of a play-out buffer starvation. If a play-out buffer starvation occurs, the heuristic downloads the next segment in the lowest quality possible and goes back into the buffering state (lines 10-12). If the download of the last segment was later than expected, the quality is decreased (lines 13-14). Thirdly, if the buffer drops below the panic threshold P, the lowest quality is downloaded as well, and the algorithm goes back into the buffering state (lines 15-17). This panic mode is to avoid the occurrence of play-out buffer starvations due to a sudden buffer under run.

If the buffer state is sufficiently high (i.e., higher than P) and no anomalies have occurred, the heuristic evaluates investigates the evolution of the buffer. If it is slowly changing, the heuristic intervenes by changing the quality level of the next segment either negatively (lines 19-20) or positively (lines 21-22). Note that, in the latter, the quality is only increased if bandwidth measurements indicate that there is enough bandwidth to download the next segment at a higher quality. If the buffer evolution is more rapid, the heuristic switches to a panic mode when the buffer is lower than L. The result is that the quality of the next segment is set to the lowest possible and the heuristic returns to buffering state (lines 23-25). Note that the algorithm compares with the lower threshold L and not with the panic threshold P as it detects a rapid change in the buffer state. If the buffer evolution is not slowly changing and not decreasing, a quality increase is again attempted (lines 26-27). In cases not identified, the quality of the next segment is not modified. Once the quality has been determined, the segment is downloaded (line 28) and the whole procedure is repeated when this segment is received.

Algorithm 1 Algorithmic details of the Microsoft's Smooth Streaming algorithm

```
    state ← BUFFERING
    while SegmentsToDownload do
      if state ≡ BUFFERING then
        if BufferDecreasing ∨ BufferSlowlyChanging then
5:        Limit change to 1 quality level
        q ← Calculate Quality According to Bandwidth
        if buffer ≥ U + L/2 then
          state ← STEADY
      else if state ≡ STEADY then
10:     if Play-out buffer starvation then
          q ← 0
          state ← BUFFERING
        else if Last download was late then
          q ← q − 1
15:     else if buffer < P then
          q ← 0
          state ← BUFFERING
        else if BufferSlowlyChanging then
          if buffer < L then
20:         q ← q − 1
          else if buffer > U then
            Attempt increasing quality
        else if BufferDecreasing ∧ buffer < L then
          q ← 0
25:       state ← BUFFERING
        else
          Attempt increasing quality
      Download at quality q
    end while
```

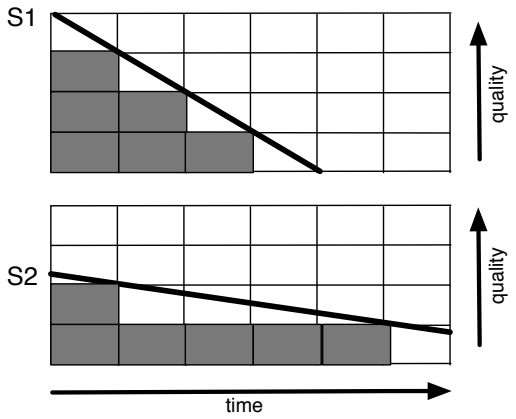


Fig. 1. Overview of the sloping-based SVC-heuristic for two slope configuration S_1 and S_2. The steeper the slope, the more enhancement layers are prioritized over subsequent base layers.

B. SVC-based heuristics

1) Naive port of Microsoft's Smooth Streaming heuristic: It is fairly straightforward to use Microsoft's Smooth Streaming heuristic for quality selection of SVC sessions as well. In this case, the quality decision is translated into the download of one or more layers. For example, if the heuristic described above dictates to download the second lowest quality level, the SVC-based variant will download the base layer and one enhancement layer of the video. The downside of this approach is that it does not fully exploit the specific characteristics of SVC-based videos. A state-of-the art algorithm, which is able to exploit these characteristics, is discussed in the next section.

2) Sloping-based SVC heuristic: An SVC-based client heuristic can allow more fine grained decisions than AVC-based client heuristics for two reasons. First, in the AVC case, a decision on the next quality to download is needed every s seconds with s being the duration of the segment. For SVC, this typically occurs a multitude of this, as decisions can be re-evaluated after every download of a layer of the segment and these layers smaller in size than the AVC-based segment themselves. Second, the AVC-based decision can only download the next segment and needs to decide on the correct quality for that segment. In contrast, the SVC-based heuristic can, at any time, decide to download the base layer of a new segment or increase the quality of a previously - not already played - segment by downloading an additional enhancement layer. Note that this does not come with an additional cost as the base-layer and lower enhancement layers are needed as well to decode the additional enhancement layer.

Andelin *et al* proposed such an algorithm in the past. We summarize it here but refer to [17] for a full explanation. The algorithm reviews the quality selection decision after the download of every layer. It can be configured to give priority to either prefetching (downloading for future segments) or backfilling (downloading for the current segments). This configuration is done by defining a slope in the heuristic: the

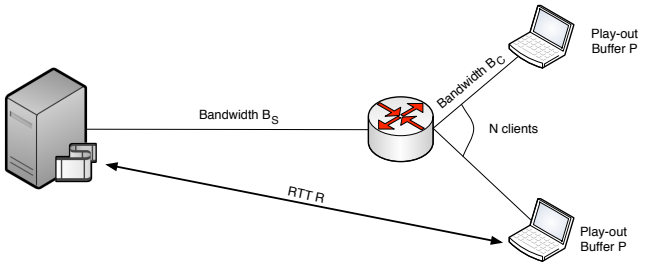


Fig. 2. Investigated network topology representing a video server, which offers a HAS-based video streaming service to N clients; The parameters N, B_s, B_c, P and R are varied to investigate different network and video player configurations

steeper the slope, the more backfilling will be chosen over prefetching. Similarly, the flatter the slope, the more additional base layers of new segments will be downloaded. This is illustrated in Figure 1, which shows the quality selection behavior for two configuration of the slope parameter S_1 and S_2. The configuration of the slope parameter of course significantly influences the behavior of the algorithm. Andelin *et al* showed that a flatter slope is needed when (1) the rate is low, (2) the rate is highly variable or (3) the rate has a high persistence but a chance of being low. In the next section, we vary the slope parameter in our comparison study of this heuristic with the two earlier described AVC- and SVC-based heuristics.

IV. Evaluation results

A. Simulation set-up

The three different video client heuristics were compared given different network and video player configurations. The goal of this study is to identify the conditions under which AVC- or SVC-based HAS performs best and to derive guidelines for configuring and deploying them. The performance of all three video client heuristics was evaluated using the NS-3 simulator[4] in combination with the Network Simulation Cradle (NSC) module[5]. NSC is a framework that allows using the operating system's actual TCP/IP stacks to perform the simulations, which significantly improves the accuracy of the results.

An overview of the used network topology is depicted in Figure 2. In the investigated network scenario, a video server provides a HAS-based streaming service to a set of N clients. Most experiments were conducted with only a single client, allowing us to characterize the behaviour of the client algorithms in detail. However, a subset of the experiments was repeated for 10 clients, in order to investigate how the algorithms handle competition for a limited amount of shared bandwidth among clients. Each video client has a play-out buffer of P seconds, which is varied from 6 up to 24 seconds to investigate both small buffer (e.g., live TV) and large buffer (e.g. Video on Demand) use-cases. Furthermore, we dynamically limit the dedicated client bandwidth B_c and shared server bandwidth B_s to introduce fixed network congestion or dynamic bandwidth fluctuations. Finally, the RTT between the clients and server is also varied, as it is inversely correlated with TCP throughput.

[4]Network Simulator 3 - http://www.nsnam.org

[5]WAND Network Research Group: Network Simulation Cradle - http://research.wand.net.nz/software/nsc.php

TABLE I. The minimum, average and maximum bitrates (in kbps) of the H.264/AVC and H.264/SVC encoded versions of the video used in the experiments

Encoding	Metric	Layer 0	Layer 1	Layer 2
AVC	Minimum	107	137	186
	Average	2087	2781	3919
	Maximum	5792	7689	10979
SVC	Minimum	109	166	261
	Average	2134	3106	4918
	Maximum	5897	8986	14507

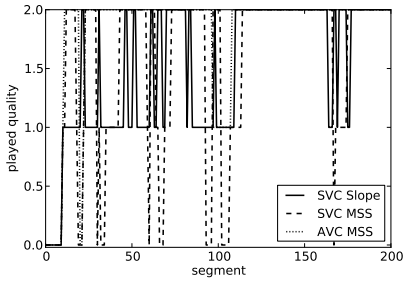
The simulations were conducted using the traces of a 400 seconds variable bitrate (VBR) video file, encoded for both H.264/AVC and H.264/SVC using the JSVM 9.19.15 Encoder. It consists of 200 segments, with an average, maximum and minimum duration of respectively 1.998, 2.533 and 1.067 seconds. All segments are available in three quality layers. Table I presents the minimum, average and maximum (cumulative) bitrates of each layer. The differences in bitrates between AVC and SVC of the base layer are due to optimizations of the encoder, while the differences of higher layers can be accounted to the encoding overhead of SVC. For the employed video, this overhead is on average 11.7% and 25.5% for layers 1 and $1+2$ respectively.

Several combinations of the MSS panic, lower and upper threshold parameters were evaluated under a variety of conditions, which showed that overall a panic, lower and upper threshold of respectively 25%, 40% and 80% of the total buffer size lead to good results. Similarly, the slope parameter of the SVC Slope heuristic was chosen to be -50%, as positive values caused significant fluctuations in requested quality.

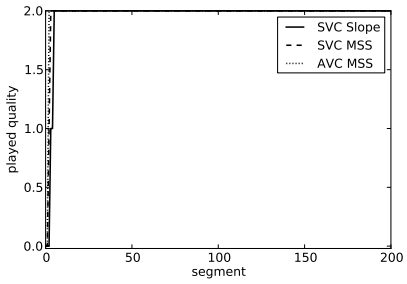
B. Behaviour over time

The study of how the algorithms adapt quality over time, under a constant available bandwidth, can provide us with insights that cannot be easily derived from results averaged over the entire run period, such as their stability and convergence behaviour. The results provided in this section depict the played video quality, specified in terms of layers (i.e., 0 for lowest quality, 1 for medium quality and 2 for highest quality), over time. The parameters B_s, R and P were fixed at 50Mbps, 10ms and 12s, as results showed they did not have any significant effect on the solution. On the other hand, the bandwidth per client parameter B_c does significantly influence results. Figure 3 compares the quality evolution of the three presented HAS client heuristics for two different values of B_c, respectively representing a scenario with congestion (cf. Figure 3a) and with enough available bandwidth for playing the highest quality (cf. Figure 3b).

Figure 3a shows the quality evolution of the three algorithms over time in a low-bandwidth scenario. All algorithms show highly erratic behaviour, changing quality continuously. Such short-term quality fluctuations are often perceived as disruptive by users and thus significantly reduce QoE. Moreover, the average quality of the two SVC-based algorithms is similar at around 1.5, although SVC Slope reverts to the lowest quality less often, it also does not reaches the highest quality as often. On the other hand, AVC MSS achieves a much higher average quality of 1.8. This is explained due to the lower overhead

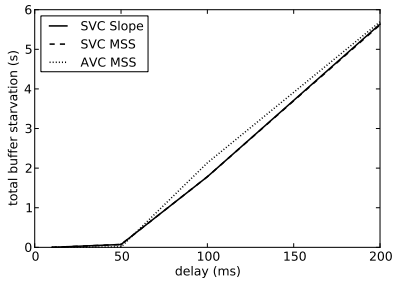


(a) $B_c = 2500$kbps

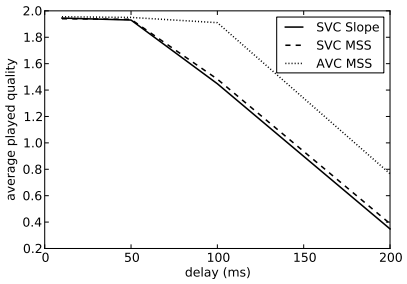


(b) $B_c = 7500$kbps

Fig. 3. The evolution of played video quality (specified in terms of layer) over time; for $B_s = 50$Mbps, $R = 10$ms, $P = 12$s and variable B_c



(a) Total buffer starvation



(b) Average played quality

Fig. 4. The total buffer starvation time and average played quality as a function of RTT (in ms); for $B_s = 50$Mbps, $B_c = 5000$kbps and $P = 12$s

of AVC encoded content. The increased overhead of SVC content thus proves to be a significant drawback in bandwidth constrained scenarios.

Figure 3b shows how the heuristics behave when bandwidth is plentiful. Although they all converge to the highest quality, the AVC MSS variant achieves it slightly faster than the SVC MSS and SVC Slope variants. This can also be attributed to the lower encoding overhead of AVC content. Specifically, the lower encoding overhead of AVC content causes the buffer to fill faster, which is often used as an indicator to increase quality by state of the art HAS algorithms. The convergence speed is especially important in use-cases where channel switching is possible (e.g., Internet Television) or under fluctuating bandwidth (cf. Section IV-D).

C. High RTT

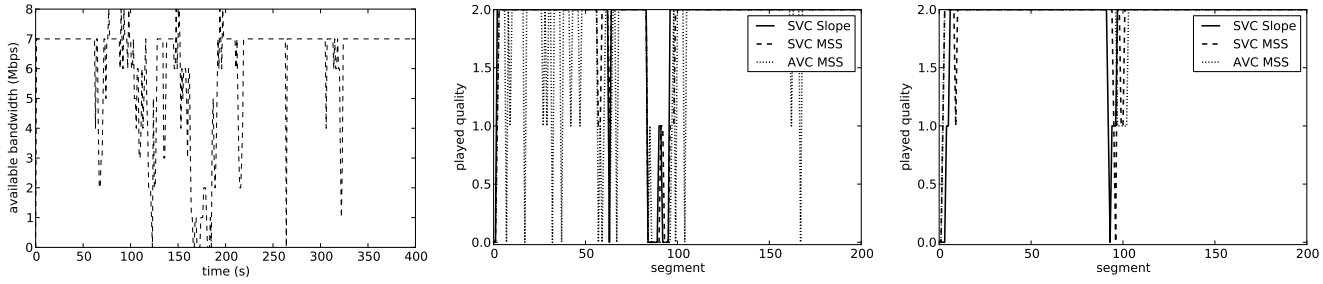
The throughput of a TCP connection is known to be inversely proportional to the RTT. This section studies the performance of the presented HAS client heuristics under these conditions. Figure 4 depicts the total buffer starvation and average played video quality as a function of the total RTT, with $B_s = 50$Mbps, $B_c = 5000$kbps and $P = 12$s. The total buffer starvation is defined as the time (in seconds) that the client application has to wait for segments it wants to play. This happens when the client buffer contains no more segments and the current segment has finished playing. The user experiences this as a freeze of the last played video frame, which is known to lead to significantly reduced QoE.

As shown in Figure 4a, an increasing RTT clearly has an increasing effect on the total length of buffer starvations. This is easy to explain: as the RTT increases, the experienced TCP throughput decreases and therefore the probability increases of not having enough throughput for streaming even the lowest quality to the HAS clients. Additionally, if the RTT increases, the idle time between sending a request and receiving the associated segment increases. While the different heuristics experience approximately the same length of buffer starvations under an increasing RTT, there are significant difference in the average played quality. This is depicted in Figure 4b. The figure clearly shows that SVC-based algorithms are much more vulnerable to high RTTs than AVC-based ones. This is mostly due to the fact that when using SVC the number of requests that need to be sent to the server is much higher, as each layer needs to be requested separately. This increases the idle time and thus significantly reduces the effective TCP throughput.

The idle time between subsequent requests can be reduced or even removed by using HTTP pipelining. This technique allows the next HTTP request to be sent before the previous segment has been fully received. As proven by the results, the use of pipelining would be especially beneficial in combination with SVC, although it would also increase performance for AVC-based services.

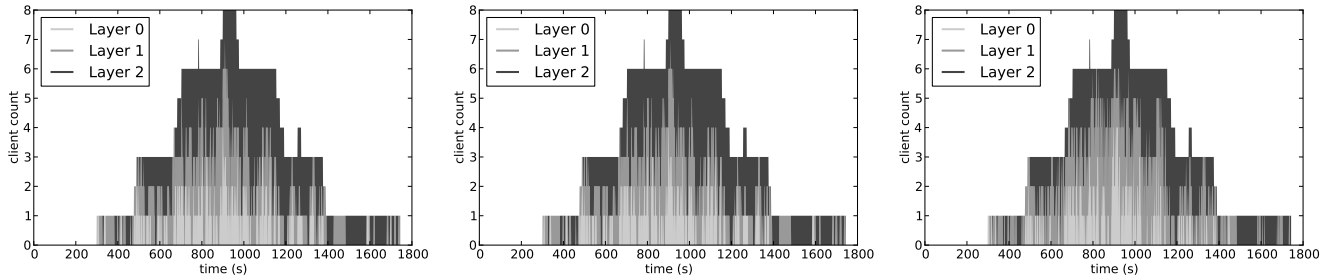
D. Bandwidth fluctuations

An important use case for HAS deployment is a mobile environment, where users watch video on handhelds such as smartphones or tablets using a mobile cellular connection. In a mobile environment, the network is more prone to bandwidth fluctuations caused by interference in the shared wireless spectrum. One of the most important advantages of HAS is its theoretical ability to cope with these bandwidth fluctuations by performing a graceful degradation of the video. In this section, we investigate the performance of all three HAS algorithms under a fluctuating bandwidth. In order to accurately model the observed bandwidth fluctuations in a mobile environment, we did a tram ride in the city of Ghent and continuously



(a) Available bandwidth B_c (b) $P = 6$s (c) $P = 24$s

Fig. 5. The evolution of played video quality (specified in terms of layer) over time under fluctuating bandwidth B_c; for $B_s = 50$Mbps, $R = 10$ms



(a) AVC MSS (b) SVC MSS (c) SVC Slope

Fig. 6. The played video quality as a function of simulation time (in s); for $B_c = 7500$kbps, $B_s = 30$Mbps, $R = 100$ms and $P = 6$s

measured the observed throughput from an UMTS connection. This measured throughput was then used in our simulation experiments to dynamically limit the client's bandwidth B_c. Figure 5a illustrates the bandwidth fluctuations observed by the client; other parameters were $R = 10ms$ and $B_s = 50Mbps$. Figures 5b and 5c show the played quality, given the observed bandwidth fluctuations, for a play-out buffer of 6 and 24 seconds, respectively.

The results show how the three algorithms react differently to the bandwidth fluctuations, especially for small buffer sizes (Figure 5b). The AVC MSS algorithm is shown to be least capable of gracefully adapting to highly variable bandwidth. The periodic drops in available bandwidth cause it to continuously switch between different qualities. On the other hand, the SVC-based are much less influenced by brief drops in throughput. The SVC Slope algorithm only needs to revert to the lowest quality during the two brief periods when the connection has temporarily failed and nothing can be downloaded for a few seconds. SVC's better performance is explained due to its ability to keep its buffer filled more easily. When using SVC, the algorithm can first download multiple base layers and then go on to higher quality layers. However, when using AVC, the entire segment needs to be downloaded as a whole. This causes the buffer to deplete at times of reduced bandwidth and cause the AVC-based algorithm to reduce the requested quality. As depicted in Figure 5c, increasing the buffer size allows all algorithms to better cope with bandwidth fluctuations. When using a buffer of 24 seconds, all algorithms can maintain the highest quality, except when severe throughput problems occur between seconds 160 and 180.

E. Network congestion

All previously presented results characterised the behaviour of a single client. However, in reality, multiple clients will compete in a shared bandwidth medium. In this section, we evaluate the heuristics' ability to fairly share the available bandwidth on a single link among multiple clients. This is done by counting the number of clients, at every instant in time, at each of the three quality layers. Optimally, the clients will be split over at most two layers, which means that the algorithm treats all clients as fairly as possible. When available bandwidth is very low or very high, all clients usually receive the minimum or maximum quality respectively. As such, we consider results with average available bandwidth, where differentiation between clients can be observed. Figure 6 depicts results for the three algorithms, with $B_c = 7500$kbps, $B_s = 30$Mbps, $R = 100$ms and $P = 6$s. The depicted results show 10 clients, their start times determined by a Weibull distribution with mean 900 seconds and shape 2.5.

The figures show that all algorithms show unbalanced behaviour, serving all three qualities simultaneously. To achieve fairness, it would be more appropriate to reduce the number of clients which receive the highest quality to be able to increase the quality of those receiving the lowest. However, due to a lack of coordination and TCP dynamics the algorithms fail to achieve such fairness.

V. Conclusions

This paper presents a quantitative comparison of H.264/AVC- and H.264/SVC-based HTTP Adaptive Streaming (HAS). Two state-of-the-art HAS client heuristics were

evaluated: the AVC-based Microsoft Smooth Streaming (MSS) heuristic and the SVC-based Slope heuristic. As a third approach, the MSS heuristic was applied to SVC-based content as well. These three heuristics were compared under a variety of network and client conditions, such as available bandwidth, round-trip time (RTT) and client buffer size.

The presented simulation results show that the differences in Quality of Experience between AVC and SVC stem from two opposing facts. First, the SVC encoding process adds significant overhead in terms of segment bitrate to higher qualities. For example, the video used in the performance evaluation added 11.7% and 25.5% overhead when delivering medium and high qualities respectively. As a consequence, more bandwidth is needed to deliver SVC-based content at the same quality as AVC-based content. Second, AVC-based algorithms need to download segments in their entirety. On the other hand, SVC-based approaches can download them layer per layer, increasing the degrees of freedom and decision making points. Thus allowing for more elaborate and intelligent client quality adaptation algorithms. Specifically, the evaluation led to the following conclusions:

- The extra SVC overhead leads to significantly reduced performance under constrained bandwidth. Concretely, SVC-based algorithms cannot achieve the same quality as AVC-based solutions when the available bandwidth is significantly lower than what is need for streaming the highest quality.
- The SVC encoding overhead additionally causes SVC-based algorithms to converge slower to the optimal quality. This is especially important for services that support channel switching, such as Internet television.
- Both evaluated algorithms showed highly erratic quality switching under constrained bandwidth. This could be improved by increasing the quality more conservatively or incorporating checks that prohibit the algorithms from frequently changing their quality. Obviously, with the latter solution, care should be taken that this does not cause unnecessary buffer starvations.
- SVC-based algorithms are far more vulnerable to high RTTs than their AVC-based counterparts. Although the amount of buffer starvation that occurs is comparable, the AVC-based algorithm could retain its quality under much higher latencies. This is due to the idle times that occur when requesting segments and could be solved by the use of HTTP pipelining.
- The SVC-based algorithms were shown to be able to cope much better with highly variable bandwidth, such as present in mobile scenarios. Brief reductions in bandwidth caused the AVC-based algorithm to immediately reduce its quality, while both SVC-based approaches were capable of retaining theirs. This is due to the fact that the AVC algorithm needs to download segments as a whole, which causes the buffer to deplete more rapidly at times of temporarily reduced bandwidth.
- It was shown that none of the evaluated algorithms are capable of balancing the quality among multiple clients, resulting in an unfair distribution of the available bandwidth. This is caused by the lack of cooperation among client instances, and could be improved by letting them collaborate or introducing intelligent bandwidth balancing components inside that network.

In summary, this article showed that AVC- as well as SVC-based HAS has its advantages and disadvantages. Specifically, AVC performs better under high latencies, while SVC more easily adapts to sudden and temporary bandwidth fluctuations when using a small buffer.

ACKNOWLEDGMENT

Steven Latré is funded by a grant of the Fund for Scientific Research, Flanders (FWO-V). This research was partially performed within the IBBT MISTRAL project (under grant agreement no. 10838). Alcatel-Lucent was partially funded by IWT project 110112.

REFERENCES

[1] Cisco Systems, "Cisco visual networking index: Forecast and methodology, 2010–2015," http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf, 2011, last accessed: 8 March 2012.

[2] R. Huysegems, B. De Vleeschauwer, T. Wu, and W. Van Leekwijck, "SVC-based HTTP adaptive streaming," *Bell Labs Technical Journal*, vol. 16, no. 4, pp. 25–41, 2012.

[3] Y. Sánchez de la Fuente, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. De Vleeschauwer, W. Van Leekwijck, and Y. Le Louédec, "iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 257–264.

[4] T. Oelbaum, H. Schwarz, M. Wien, and T. Wiegand, "Subjective performance evaluation of the svc extension of h.264/avc," in *15th IEEE International Conference on Image Processing (ICIP)*, 2008, pp. 2772–2775.

[5] R. Pantos and W. May, "HTTP Live Streaming," 2011. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-07

[6] T. Stockhammer, "Dynamic adaptive streaming over http - standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, ser. MMSys '11, ACM. New York, NY, USA: ACM, 2011, p. 133–144. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943572

[7] L. De Cicco, S. Mascolo, and C. Abdallah, "An experimental evaluation of akamai adaptive video streaming over hsdpa networks," in *Computer-Aided Control System Design (CACSD), 2011 IEEE International Symposium on*, sept. 2011, pp. 13 –18.

[8] O. Oyman and S. Singh, "Quality of experience for http adaptive streaming services," *IEEE Communications Magazine*, vol. 50, no. 4, pp. 20–27, april 2012. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2012.6178830

[9] C. Liu, I. Bouazizi, and M. Gabbouj, "Parallel adaptive http media streaming," in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, 31 2011-aug. 4 2011, pp. 1 –6.

[10] W. Pu, Z. Zou, and C. W. Chen, "Dynamic adaptive streaming over http from multiple content distribution servers," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1 –5.

[11] ——, "New tcp video streaming proxy design for last-hop wireless networks," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, sept. 2011, pp. 2225 –2228.

[12] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1103 –1120, sept. 2007.

[13] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. D. Vleeschauwer, W. V. Leekwijck, and Y. L. Louédec, "Efficient http-based streaming using scalable video coding," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 329 – 342, 2012, ¡ce:title¿Modern Media Transport – Dynamic Adaptive Streaming over HTTP (DASH)¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596511001147

[14] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over http in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288 – 311, 2012, ¡ce:title¿Modern Media Transport – Dynamic Adaptive Streaming over HTTP (DASH)¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596511001135

[15] V. Adzic, H. Kalva, and B. Furht, "Optimized adaptive http streaming for mobile devices," in *Applications of Digital Image Processing XXXIV*, A. G. Tescher, Ed., vol. 8135, no. 1, SPIE. SPIE, 2011, p. 81350T. [Online]. Available: http://link.aip.org/link/?PSI/8135/81350T/1

[16] T. Schierl, Y. Sanchez de la Fuente, R. Globisch, C. Hellge, and T. Wiegand, "Priority-based media delivery using svc with rtp and http streaming," *Multimedia Tools and Applications*, vol. 55, pp. 227–246, 2011, 10.1007/s11042-010-0572-5. [Online]. Available: http://dx.doi.org/10.1007/s11042-010-0572-5

[17] T. Andelin, V. Chetty, D. Harbaugh, S. Warnick, and D. Zappala, "Quality selection for dynamic adaptive streaming over http with scalable video coding," in *Proceedings of the 3rd Multimedia Systems Conference*, ser. MMSys '12, ACM. New York, NY, USA: ACM, 2012, p. 149–154. [Online]. Available: http://doi.acm.org/10.1145/2155555.2155580